The Thesis Committee for Oriana Yuen-Quan Wong
Certifies that this is the approved version of the following thesis:

Bi-Directional Dc-Ac Inverter

APPROVED BY

SUPERVISING COMMITTEE:

Surya Santoso, Supervisor

Mark Flynn

Bi-Directional Dc-Ac Inverter

by

Oriana Yuen-Quan Wong

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2018

This work is dedicated to my mother who has supported me every step along the way and my father who first introduced me to engineering.

Acknowledgments

I would like to express my gratitude to my advisor, Dr. Santoso, for his guidance and the opportunity to work in the ASPIRE lab. I would also like to thank Dr. Flynn for his guidance and for reading this thesis. I gratefully acknowledge Alvaro Furlani Bastos, Keng Weng Lao, and Piyapath Siratarnsophon for the insightful discussion and advice while I was completing this work.

Abstract

Oriana Yuen-Quan Wong, M.S.E.
The University of Texas at Austin, 2018

Supervisor: Surya Santoso

Photovoltaic (PV) installations in the distribution grid have increased in recent years due to technological improvements in power electronics and supportive governmental tax incentives. However, the abundant growth of distributed solar raises general concern over overvoltage and reverse power flow on the distribution grid. Battery energy storage systems and smart inverters have been proposed to address these problems by absorbing the reverse power flow and providing voltage-reactive power support (also known as volt-var control) to the grid. This work details the design, analysis, and construction of a controllable bi-directional inverter that can connect batteries to the distribution grid and perform voltage-reactive power support. The digital control system is implemented on a TI TM4C123GXL launchpad with a voltage-source H-bridge inverter. First, the supporting modules for the inverter, such as the measurement circuitry, grid synchronization code, and the sinusoidal wave generation method, are all described and verified separately. To test the inverter controller, the inverter uses two 12 V/12 Ah batteries as the dc source and is connected to a tabletop distribution grid model. For the volt-var support test, the controller indirectly calculates the required reactive power to perform voltage regulation and operates the inverter to restrain the voltage at the point of common connection to be within $\pm 2\%$ of the target voltage. For the bi-

directional power flow test, the inverter operates the battery between the charging and discharging modes.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Growth of Photovoltaics on the Grid

Distribution-side solar, such as community and residential solar, continues to steadily grow in the US, especially in California and New York where they are targeting an aggressive renewable portfolio standard (RPS) of 50% by 2030 [1]. In other words, by 2030, at least 50% of the electricity generated by in-state utilities will be from renewable resources. In fact, the Solar Energy Industries Association (SEIA) forecasts that the total installed photovoltaics (PV) in the United States will continue to grow over the next five years regardless of any political tariffs [2]. Although solar generation is a popular and environmentally-friendly energy source, abundant growth of PV penetration can lead to overvoltage and reverse power flow on the distribution grid [3,4]. For example, reverse power flow occurs when the PV generation is higher than the load consumption. In this case, the excess generated power causes a voltage rise in the distribution network.

1.2 Motivation for a Bi-directional Inverter

A proposed solution to address the adverse effects of distributed PV generation growth is to deploy grid-scale batteries to absorb reverse power flow during high PV power output and supply the power back when PV generation drops out [5]. With grid-scale batteries, the connected power inverter needs to handle bi-directional power flow to charge and discharge the battery. Since

inverters are already required to connect batteries to the grid, it is advantageous for them to also provide local voltage correction through reactive power control.

This work details the design, analysis, and construction of a controllable bi-directional inverter that can connect batteries to the distribution grid and meet the volt-var requirements described in Table 1.1 where the regulated voltage needs to be within $\pm 2\%$ of V_{ref} and the response time is less than 90 s.

Parameters	**Description**	**Range**
Voltage-Reactive Power Mode Enable	Enable voltage-reactive power mode	Yes/No
Deadband (d)	The range of tolerance around V_{ref} where no control action occurs	$\pm 2\%$
Open Loop Response Time	Time to ramp up to 90% of the new reactive power target in response to the change in voltage	1s to 90s

Table 1.1: Voltage-Reactive Power Mode Requirements

For the inverter's construction, the digital control system is implemented on a Texas Instrument TM4C123GXL interfacing with a voltage-source H-bridge inverter. First, the supporting modules for the bi-directional inverter are all identified, designed, and verified separately. These modules include the measurement circuitry, the grid synchronization code, and the sinusoidal PWM methods. The inverter is then used to connect the battery energy storage to a tabletop distribution grid model and tested to perform volt-var control. During the experimentation, the inverter successfully charged the battery from 12.4 to 12.8 V. The inverter controller also successfully regulated the voltage to stay between 9.8 V and 9.9 V during steady-state. This range is within $\pm 2\%$ of 10 V, the chosen V_{ref} for this work. The bi-directional inverter provides future researchers a controllable inverter that can be integrated into a microgrid or as a starting platform to test more smart inverter features.

1.3 Thesis Outline

This thesis consists of five chapters and is organized as follows:

Chapter 1 introduces the potential problems increased PV penetration has on the grid, the proposed solution of energy storage and smart inverters, the volt-var control requirements of the work, and the motivation behind building a bi-directional inverter.

Chapter 2 discusses the construction of the main H-bridge circuit, the data acquisition of the inverter controller, the inverter's method of generating an output sinusoidal wave, and the code for grid synchronization. The chapter also discusses the design of a signal conditioning board for ac signals, a simple voltage follower circuit for dc measurements, and the isolation circuit between the MOSFET gate driver and the microcontroller output pins.

Chapter 3 explains the overall control algorithm of the inverter controller, the required parameters for the inverter to transition to the battery charging mode, and the PI control loop of the volt-var regulation.

Chapter 4 presents the validation of the data acquisition, sinusoidal output, and the grid synchronization of the inverter. The chapter then describes the tabletop distribution grid model setup and how the bi-directional inverter connects to the system. Lastly, the charging capability of the inverter and the voltage regulation method are tested and analyzed.

Chapter 5 concludes the thesis and discusses future experiments that can add additional smart functions to the inverter or incorporate the inverter into a larger system like a microgrid or a distributed energy resource management system.

Chapter 2

Inverter Setup

In this chapter, the design of the main H-bridge, measurement circuits, and controller software modules is discussed. The main power circuit of the inverter is a voltage-source H-bridge that can function as a dc-ac power inverter or as a passive ac-dc rectifier depending on the proper control signals. There are three important functions a controller needs to have to transform the H-bridge into a controllable power inverter. The controller needs to have the ability to output a sine wave, perform feedback control, and synchronize the inverter output to the grid regardless of wave distortions from the load [6]. The controller, a Texas Instruments TM4C123GXL (TM4C) microcontroller, performs all the aforementioned tasks by using a unipolar SPWM technique to output the sine wave, the ADC to sample all input ac and dc signals before performing control actions, and a phase-lock loop to synchronize the inverter voltage reference with the grid voltage. The supporting circuitry for these functions is described in their respective sections.

2.1 H-Bridge Circuitry

The H-bridge, shown in Figure 2.1, is a generic circuit configuration that can be used for any application that requires current to flow in opposite directions such as a dc-ac power inverter or a motor controller [7]. There are three different switching sequences for the H-bridge output:

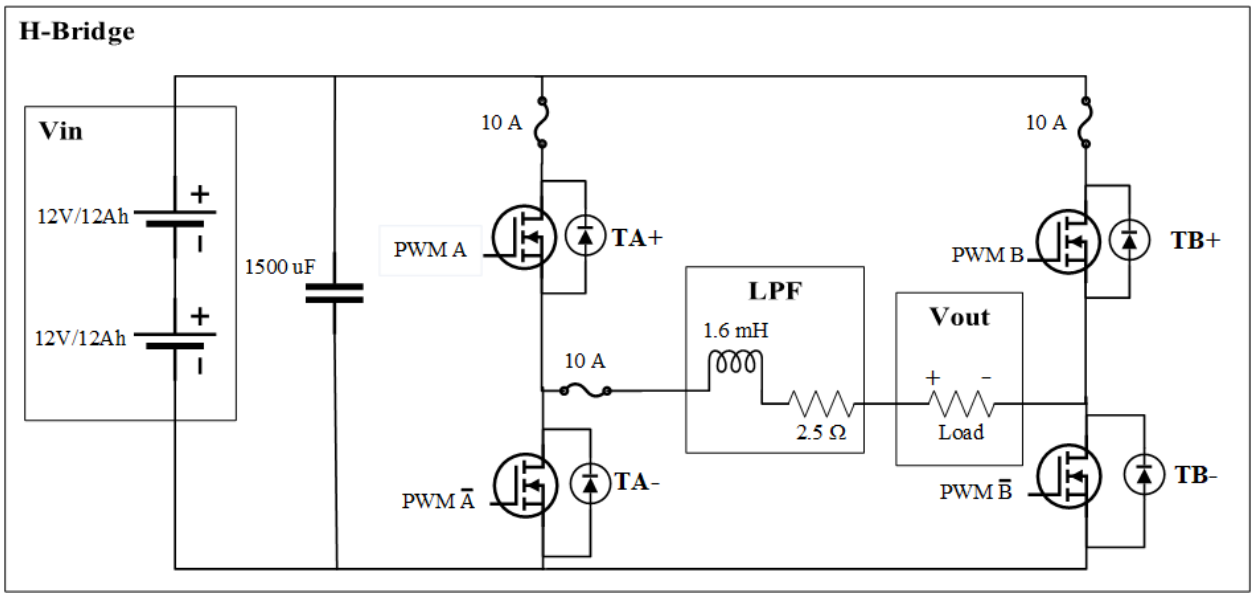


Figure 2.1: H-bridge as a Power Inverter

- When TA+ and TB- are turned on, $V_{out} = +V_{in}$ V
- When TB+ and TA- are turned on, $V_{out} = -V_{in}$ V
- When (TA+ and TB+) or (TA- and TB-) are turned on together, $V_{out} = 0$ V

In this work, the H-bridge is built with four Metal Oxide Semiconductor Field Effect Transistors (MOSFETs) connected to two 12 V/12 Ah batteries for V_{in} and a RL low-pass filter at the output. When the gating signals are provided, the H-bridge acts as a single-phase power inverter. Otherwise, the circuit takes advantage of the four MOSFETs' body diodes as a diode bridge and behaves as a passive ac-dc rectifier. For the RL output filter, a 1.6 mH inductor and 2 Ω resistor are chosen to pass only the fundamental 60 Hz voltage through. Any harmonics at higher frequencies

than the filter's cutoff frequency of 241.15 Hz is filtered out. This low cutoff frequency allows future users to implement slower switching frequencies. For this work, the switching frequency is set to 6 kHz. The filter introduces a phase shift of -13.5 °at 60 Hz that is accounted for in the microcontroller code.

2.2 Data Acquisition

Proper measurement and calculation of the physical circuit parameters is vital to interface the power hardware to the controller. Inaccurate measurements can lead to incorrect control actions that can harm the inverter or microcontroller. The first step of proper measurement is to condition the input ac or dc signals. Since the TM4C analog-digital converter (ADC) ports only accept voltages from 0 V to 3.3 V, supporting circuitry is necessary to drop the incoming signals to appropriate voltage levels [8]. For ac measurements, a signal conditioning board is used to decrease, level shift, and filter the incoming ac voltage and current signals. For dc measurements, a voltage divider with a voltage follower is used to decrease the dc voltage to acceptable levels.

2.2.1 Signal Conditioning Board

The ac signal conditioning board (SCB) accepts two inputs (the ac signal and +12 V_{dc}) and outputs the processed ac signal and 3 types of dc voltage (+3.3, +12, and -12 V). The SCB contains 5 stages to condition the incoming ac signal: a voltage divider, voltage follower, level shifter, inverting op-amp, and final Sallen-Key filter. The signal conditioning process flow is shown in the blue boxes in Figure 2.2 while the red boxes show the voltage conversions that happen on the board. The overall SCB concept was borrowed from [9], but improvements were made to the design. The improvements were unifying the grounds with a ground plane to prevent common-

mode coupling, re-designing the Sallen-Key filter to have no phase shift, and consolidating all the voltage inputs into one main voltage input from the wall-wart.

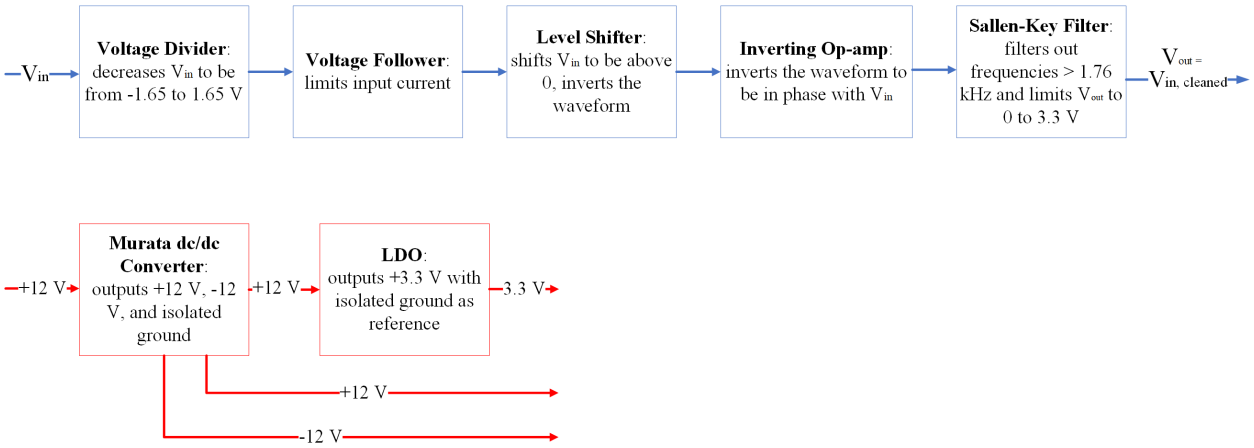


Figure 2.2: Signal Conditioning Board Process Flow Diagram

2.2.1.1 Voltage Conversion

There are two methods of powering the SCB. The first method requires the wall-wart, dc jack, and the Murata NMH1212SC dc/dc converter [10]. The wall wart plugs into the dc jack on the board and rectifies the supply ac grid voltage to +12 V dc voltage and ground reference. The Murata NMH1212SC then converts the input +12 V from the wall-wart to $\pm$12 V and an isolated ground that is separate from the ground of the wall-wart [10]. The second method requires a voltage input of +12 V, -12 V, and a ground reference, but does not use the dc jack or dc-dc converter. This method requires the input voltages to be plugged into the male header pins labeled as SV2 on the lower left hand corner of the schematic in Figure 2.3. This allows for several SCBs to be daisy chained together and reduces the number of wall-warts and dc-dc converters required to measure the system voltage and current values. For example, if SCB 1 already has the Murata

and dc-jack combination, then SCB 2 can just use the voltage output of SCB 1 to power itself. If the user does daisy chain several SCBs together, the user needs to be cautious that the load does not exceed the output current of the Murata dc-dc converter of ± 83 mA [10]. In this work, the daisy chain worked for 2 SCBs together, but due to the distance of the measurement points, the daisy chain configuration was not kept.

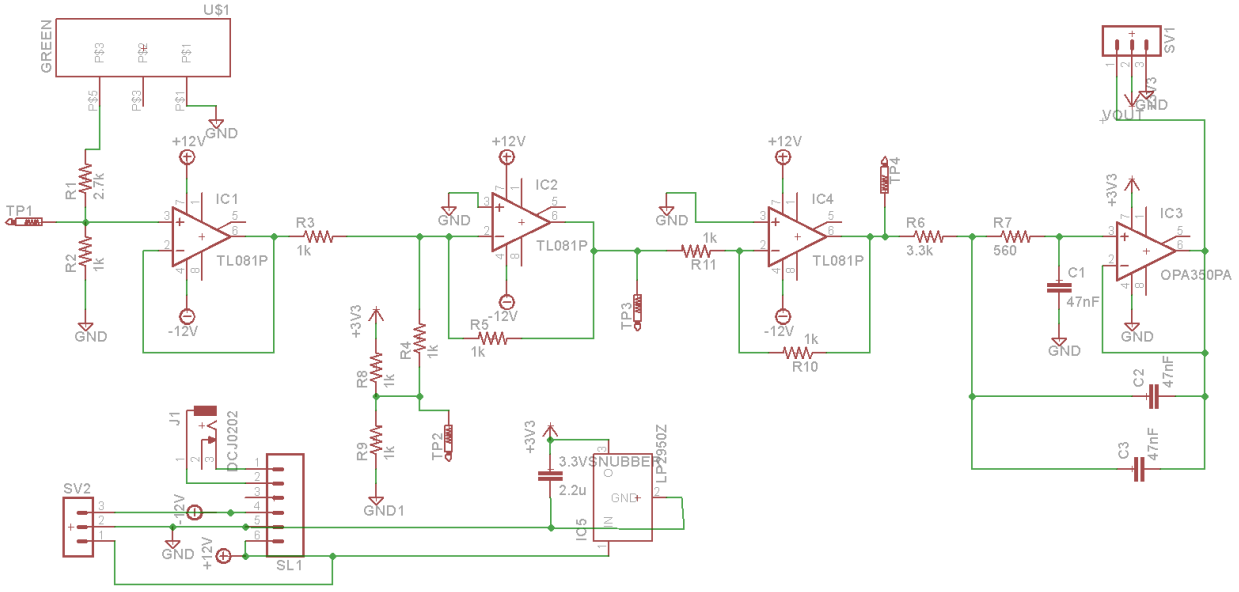


Figure 2.3: Ac Signal Conditioning Board Schematic

In order to power the operational amplifier (op-amps) and linear dropout regulator (LDO), ± 12 V is required. All the op-amps, except the last op-amp, use ± 12 V as the voltage rails to avoid clipping the op-amp's input signal. The LDO regulates the 12 V to 3.3 V. The 3.3 V is used as the voltage rail of the Sallen-Key filter and to create a 1.65 V dc offset for the summing amplifier. Originally, the LDO had a large fluctuation in its output ranging from 2.5 V to 4.2 V, but a 2.2 μF snubber capacitor was inserted at the output of the LDO to reduce the fluctuation to a range of 3.2 V to 3.36 V. Any capacitor larger than 2.2 μF had no noticeable impact on the LDO fluctuation.

The 1.65 V dc offset comes from a voltage divider with a ratio of 0.5 and the 3.3 V from the LDO. A noteworthy caveat is that while the design calls for two 1 kΩ to get 1.65 V, the physical design requires a 1 kΩ resistor in series with the 8.2 kΩ resistor due to the PCB resistance and op-amp input impedance affecting the voltage divider.

2.2.1.2 Signal Conditioning

The first stage of the signal conditioning portion of the SCB is the voltage divider that lowers V_{in} to a range of -1.65 V to 1.65 V. The decreased signal is fed into a unity-gain voltage follower. The voltage follower is seen by the input as high impedance which limits the current into the SCB and microcontroller ADC port. The next stage is the level shifter where a summing amplifier adds the incoming ac signal with a 1.65 V dc offset. The dc offset ensures the output ac signal is always positive. The last two stages are an inverting op-amp and a Sallen-Key low-pass filter. Since the input signal was inverted by the summer amplifier in the second stage, another inverting amplifier is required to invert the signal to be back in phase with the input signal. Afterwards, the 60 Hz voltage is passed into a Sallen-Key low pass filter with a cutoff frequency of 1.761 kHz. As an extra safeguard, the voltage rail of the Sallen-Key op-amp is maintained between 0 V and 3.3 V. If there are any input voltage spikes, the last op-amp will saturate the signal and clip V_{in} to fit between the voltage rail range as shown later in Chapter 4. The physical SCB is shown in Figure 2.4. During implementation, the second resistor of both voltage dividers was not soldered directly into the board. Instead, they connect into female header pins. This allows the user to easily change the voltage divider ratio when necessary such as when the SCB needs to be placed in a new position or the incoming voltage range changes.

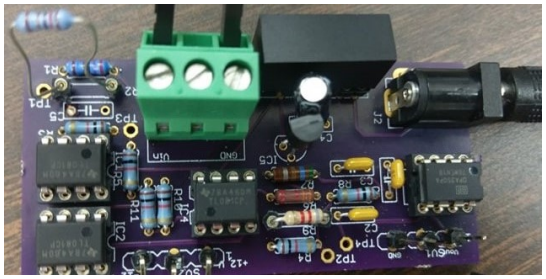


Figure 2.4: Physical Signal Conditioning Board

2.2.2 DC Measurement Circuit

The combination of a voltage divider and a voltage follower is used to measure the inverter's input dc voltage. For the voltage divider, the ratio is 8.48 where the max input is 28 V and the max output is 3.3 V as shown in Figure 2.5. Since the microcontroller is a low-impedance load,

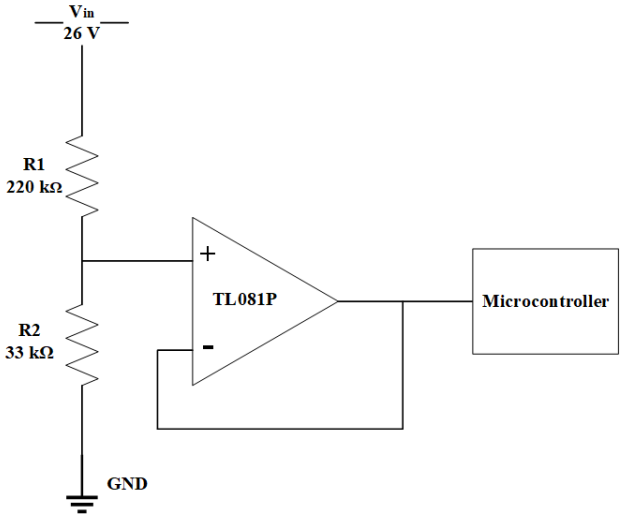


Figure 2.5: Simple Voltage Follower Circuit

the ADC port draws high current from the voltage divider. Similar to the SCB's first stage, a voltage follower is used to limit the current draw. The same Texas Instruments TL081 op-amp is used

in this circuit, but the voltage rail is supplied by a separate 3.3 V external dc voltage supply because the 3.3 V output from the SCBs was physically too far from the dc measurement point [11]. The ac and dc measurement circuits all share the same ground reference, but this is not a problem because the potential transformers (PT) for the ac measurements isolate the grid neutral from the battery's ground reference.

2.2.3 Microcontroller Sampling Configuration

In this section, the characteristics of the TM4C ADC and this work's specific configuration of the ADC are described. The ADC on the TM4C is a 12-bit converter with a resolution of 0.805 mV/bit. The sampling frequency is set to 6 kHz to avoid aliasing since 100 sample points is used to reconstruct the 60 Hz signal. For the dc voltages, a software averaging method was used to smooth fluctuations in the ADC reading. The voltages were kept in fixed point for easier data storage and reduced use of memory.

The TM4C has two modes of sampling: 1) single-ended where the reference is the microcontroller ground and 2) differential ended where the reference is the corresponding ADC pin in the differential pair [8]. For the voltage signals, the ADC is configured to be single-ended where the input voltage to the ADC pins is in reference to the microcontroller ground. This configuration works because all the dc and ac measurement circuits have their ground references tied together. However, for the current measurements, the ADC configuration for these ports are in differential mode because the current measurements are through current transformers (CT) and do not share the same reference point as the voltage measurements.

Lastly, before the ADC raw values can be used for calculation, the samples need to be converted from ADC counts to the voltage values they represent in the physical system. The

conversion equations are shown in (2.1), (2.3), and (2.1).

$$V_{ac} = \frac{((V_{sample} - 2048)3300)}{4095} K_{divider} K_{transformer} \tag{2.1}$$

$$V_{dc} = \frac{(V_{sample})3300)}{4095} K_{divider} \tag{2.2}$$

$$I_{ac} = \frac{((V_{sample} - 2048)2V_{diff})}{4095} K_{divider} \tag{2.3}$$

The subtraction of 2048 in Equation (2.1) is the fixed point hexadecimal representation of 1.65 V to account for the level shift of the SCB. During implementation, the SCB level shifter varied from 1.6 V to 1.68 V due to fluctuations from the LDO output voltage, so the 2048 was replaced with a moving mean of the incoming ac waveform. In order to scale the value back to the measured voltage, the sample is first multiplied by the voltage difference between the ADC pin and its reference, mapping the sample along the voltage range. The sample is then divided by 4095 to convert the sample from bits to voltage. For example, 4095 is the max sample value the microcontroller can sample. By multiplying by 3300 and dividing by 4095, the microcontroller knows that 4095 corresponds to 3.3 V. For all single-ended ADC samples, the voltage range is 3.3 V while the voltage range is V_{diff} for differential ADC signals. V_{diff} is the voltage difference between the positive and negative ADC pins of the differential pair.

2.2.4 Root Mean Square Calculations

Before the inverter can take any control actions, the controller needs to know the system's updated rms voltage, rms current, and power values.

$$V_{rms} = k_u \sqrt{\frac{1}{N}\sum_{i=1}^{N} u_i^2}$$

$$I_{rms} = k_i \sqrt{\frac{1}{N}\sum_{i=1}^{N} i_i^2} \tag{2.4}$$

$$P = \frac{k_u k_i}{N}\sum_{i=1}^{N} u_i i_i$$

The set of equations shown in 2.4 is used to calculate the rms voltage, rms current, and power values from the incoming instantaneous samples [12]. For the voltage and current rms values, the instantaneous samples are level-shifted back to the -1.65 and 1.65 numerical range, squared, and added into a moving sum during the ADC interrupt service routine (ISR). The power calculation multiplies the level-shifted instantaneous voltage and current before adding the result to its own moving sum. This saves the main program from running a while loop to calculate these values and being interrupted during the process. The division of the array size (N) is done in the main loop after the sampling arrays are full of 300 samples. The number 300 was picked in order to filter the ac measurements from sampling noise since the resulting rms is an average of 3 cycles of the ac waveform. The results of the division is sent to a custom fixed point square root routine which uses a simple binary search to get the answer. The rms calculations are all done with fixed point integers (ints) to save memory and time as opposed to using doubles to store everything. The true representation of these values are 3 decimal points to the right, for example 3000 in the code

represents 3 V in the physical system. Lastly, the values are multiplied by the scaling factors of the transformer ratio and voltage divider to arrive at the electrical system-level values. These rms values are then used in the overall control algorithm.

For better organization of the variables in code, the ac power values and rms values use the struct data type as shown in the code below. A struct groups a list of variables under one name and places them in the same block of memory. Instead of creating multiple variables such as ”Vinst”, ”Vrms”, ”Vsum” by name, the user can call the struct variable and select the corresponding variable inside the struct. Although real power values do not have rms values, the same struct type was used for the real power variables because the power calculation in code is similar to the rms calculation. Each ac measurement point also has its own struct named acPower that consists of voltage, current, real power, reactive power, and apparent power called acPower. Again, this saves all the relevant circuit information under one name such as ”Sinv” instead of having multiple circuit variables for the same measurement point.

Listing 2.1: Organization of rms and measurement point variables

```
struct acValue{
  int inst;
  int sum;
  double rms;
  int mean;
  int avg_sum;
};
typedef struct acValue acValues;

struct ACPower{
  acValues V;
  acValues I;
  acValues P;
  double Q;
  double S;
```

```
};
typedef struct ACPower ACPower_t;
```

2.3 Sinusoidal Pulse-width Modulation and Gate Driver

In order to replicate a sinusoidal wave, a unipolar sinusoidal pulse-width modulation (SPWM) scheme is used to switch between V_{dc}, $-V_{dc}$, and 0 V [13]. In SPWM, the input sinusoidal signal(s), V_{cont}, is compared to a triangle wave with a frequency modulation index (m_f) at least 10 times faster than the incoming sinusoidal signal. This is done in order to accurately capture the sine wave through the waveform comparisons. By comparing the input sine wave to the triangular carrier wave (V_{tri}), PWM signals are generated for each leg of the H-bridge. The inverter output voltage is the difference between leg A and leg B ($V_{out} = V_{ab} = [V_{an} - V_{bn}]$). There are two methods of SPWM: unipolar and bipolar. In bipolar, there is only one input sinusoidal signal compared with the triangular wave. Two complementary gating signals are generated from the comparison and fed into the four switches where each leg has one switch turned off while the other is on. Thus, the output V_{ab} is bipolar and always between $+V_{dc}$ or $-V_{dc}$.

For unipolar, the output is between $+V_{dc}$, 0 V, or $-V_{dc}$. The trade-off is that the unipolar scheme requires two input sine waves with the same magnitude and frequency, but 180 $^{\circ}$out of phase with each other (V_{cont} and $-V_{cont}$). The unipolar SPWM compares both inputs, V_{cont} and $-V_{cont}$ with V_{tri} and the switching rules are summarized in Figure 2.6, an image borrowed from [13] with permission from Dr. Mark Flynn. The unipolar has the advantage over bipolar SPWM because the bipolar method switches all four switches simultaneously while the unipolar switches the two switches on both legs asynchronously, slightly reducing switching losses and improving the harmonic performance of the inverter [7].

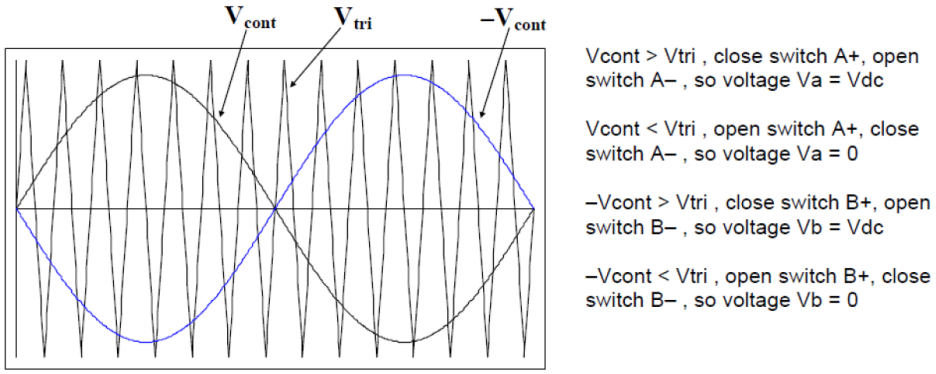


Figure 2.6: Unipolar Switching

In this work, the microcontroller achieves the unipolar scheme by sampling V_{cont}, generating $-V_{cont}$ in code, and comparing the samples to an internal counter that acts as V_{tri}. V_{tri} is a timer interrupt that interrupts at a frequency of 6 kHz. V_{cont} is at a frequency of 60 Hz, so the frequency ratio, m_f, is at 100. When generating the PWM signals, $-V_{cont}$ and V_{cont} is normalized to the same magnitude as V_{tri} before comparison. The modulation index (m_a) of the inverter output is modified during the volt-var algorithm which is further discussed in Chapter 3.

In reality, MOSFETS do not turn on and off instantaneously due to the discharge rate of the gate capacitance. This is a problem because the switches in an H-Bridge leg (TA+ and TA-) or (TB+ and TB-) can never be turned on together. If they are on at the same time, a shoot-through is created and shorts out the voltage supply. Deadtime is a time period when both MOSFET gating signals in the same leg are set to 0. This deadtime needs to be inserted between the switching pulses to avoid shoot-through as the MOSFETs in each leg turn on/off. The FDA59N25 MOSFETS has a max of 190 ns turn off time and requires V_{gs} to be at least 3 V [14]. With a higher V_{gs}, MOSFET turns on/off faster and with less losses. Also, the FDA59N25 transistors are N-channel enhancement MOSFETs, so the high side gate signal has a different reference than the low side

gate signal. A IRS21844 H-bridge gate driver is used in each leg to boost the PWM signals from 0 V to 12 V, insert an average deadtime of 400 ns, and provide a high side reference through a bootstrap capacitor circuit [15]. As extra current protection from the inverter to the microcontroller, the microcontroller PWM output ports are isolated from the inverter's gating signal input header pins through an optocoupler as shown in Figure 2.7. Since the optocoupler inverts the incoming

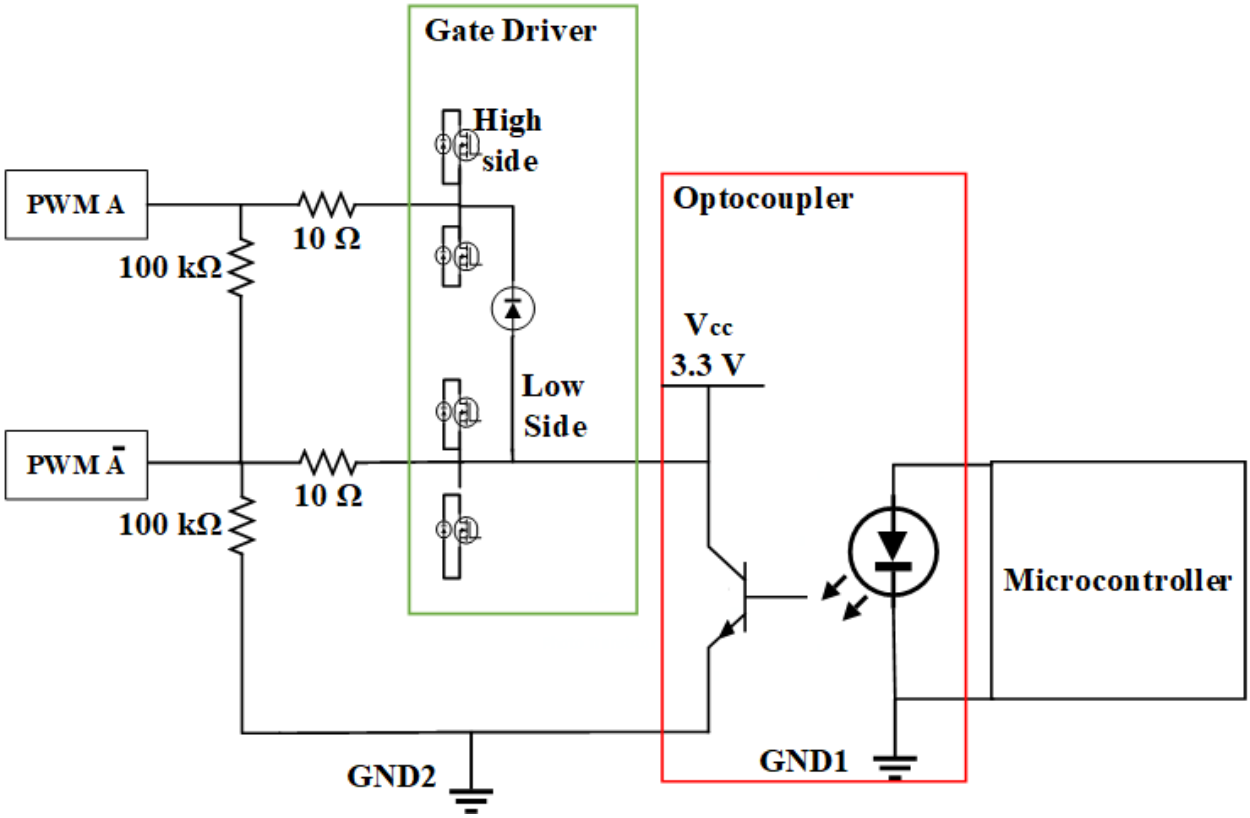


Figure 2.7: Isolated Gate Driver

signal, the output of the optocoupler is the inverse of the microcontroller output. For example, in the code, PB7 outputs the PWM for leg B, but after the inversion from the optocoupler, the

signal from PB7 is routed to leg A instead. The same circuitry is repeated for Leg B as well. It is also important to note that the input voltage source and ground of the optocoupler comes from an external dc source and the optocoupler ground (GND2) is different from the microcontroller and measurement circuit grounds (GND1). This isolation circuit was implemented after several observations of current backfeeding from the inverter circuit into the microcontroller PWM ports when the grid-tied switch was being closed. This current caused the microcontroller to halt the program and reset itself. After the physical inverter gating signal input header pins was isolated from the microcontroller pins, this problem did not happen again.

2.4 Phase Locked Loop

The importance of the phase locked loop (PLL) in this work is to synchronize the inverter voltage output (V_{inv}) with the voltage (V_{PCC}) at the point of common coupling (PCC). The PLL

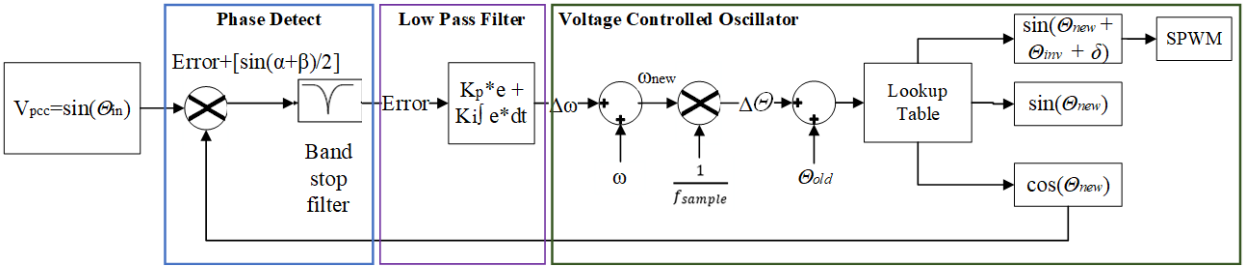


Figure 2.8: PLL Block Diagram

block diagram is shown in Figure 2.8 and has three stages: a phase detector (PD) highlighted with the blue box, a low pass filter (LPF) highlighted in purple, and a voltage controlled oscillator (VCO) highlighted in green. First, the phase difference is calculated with the $sin(\alpha)cos(\beta) = \frac{sin(\alpha-\beta)}{2} + \frac{sin(\alpha+\beta)}{2}$ trigonometric identity. In Equation 2.5, the α becomes $\omega_{PCC}t + \theta_{PCC}$ and β becomes $\omega_{PLL}t + \theta_{PLL}$.

$$V_{PCC}sin(\omega_{PCC}t + \theta_{PCC})cos(\omega_{PLL}t + \theta_{PLL}) =$$
$$V_{PCC}(\frac{sin[(\omega_{PCC}t - \omega_{PLL}t) + (\theta_{PCC} - \theta_{PLL})]}{2} + \frac{sin[(\omega_{PCC}t + \omega_{PLL}t) + (\theta_{PCC} + \theta_{PLL})]}{2}) \tag{2.5}$$

In this equation, the difference between $\omega_{PCC}t$ and $\omega_{PLL}t$ is a small value that can be ignored at steady-state because the reference angular frequency of both waveforms is 120π rad/s. After dropping $(\omega_{PCC}t - \omega_{PLL}t)$ from the equation, the first term becomes the phase difference. However, the equation has a second term that is a value at twice the reference frequency (59.95 to 60.05 Hz) range. A first order band-stop filter is used to filter out the second term by stopping any signals with frequencies between 115 and 125 Hz from passing through. The discrete band-stop filter's transfer function is shown in Equation 2.6 where Y[z] represents the output value of the band-stop filter and X[z] represents the input to the filter. X[z], in this work, is the result of Equation 2.5 with the phase difference and extra term. The B and A array of terms are the transfer function's coefficients. In this work, the Matlab Butterworth function was used to calculate the coefficients. The results for the B array is [0.9948, -1.9739, 0.9948] and for the A array is [1.0000, -1.9739, 0.9896]. Notice that the A1 and B1 pair and the B0 and B2 pair have the same value.

$$\frac{Y[z]}{X[z]} = \frac{B0 + B1z^{-1} + B2z^{-2}}{A0 + A1z^{-1} + A2z^{-2}} \tag{2.6}$$

Since this is a discrete filter, future values of the input and outputs are unknown, so the filter relies on historical samples of these waveforms. The z^{-n} terms represent the nth sample back in history of the waveform it is multiplying with. For example, in the filter equation 2.6, when you cross-multiply the two fractions, the top is multiplied by the samples from the phase detect stage and the bottom is multiplied by the band-stop filter outputs. The $B1z^{-1}$ term means that the B1 coefficient

is multiplied by the past sample of the phase detect stage from the previous iteration of the PLL code. For the $A2z^{-2}$ term, the coefficient is multiplied by the past calculated band-stop filter output from two iterations ago. Equation 2.7 shows the result of the filter output for this iteration of the PLL code after cross multiplication and the equation is solved out for Y[0]. In the code shown in [16], the arrays X and Y are named Upd and ynotch respectively.

$$A0(Y[0]) = (B0X[0] + B1X[1] + B2X[2]) - (A1Y[1] + A2Y[2]) \tag{2.7}$$

After the band-stop filter filters out the extra term, only the phase difference information is left. The inverter controller then takes this phase difference and shifts the phase of the internal sine wave until the phase difference becomes 0 in steady-state. A proportional integral (PI) controller gradually corrects the error between the phases, acts as a digital low pass filter to prevent any high frequency harmonics to pass through, and translates the phase difference to ω. In this work, the PI filter in PLL code uses the trapezoidal method for discretization where the s term in the PI transfer function, shown in Equation 2.8, is replaced by $\frac{2(z-1)}{T(z+1)}$. The input to the PI loop is the output of the band-stop filter.

$$\frac{PIout[z]}{Y[z]} = K_p + \frac{K_i}{s} \tag{2.8}$$

After substitution, the equation becomes

$$\frac{PIout[z]}{Y[z]} = \frac{\frac{2K_p+K_iT}{2} - \frac{2K_p-K_iT}{2}z^{-1}}{1 - z^{-1}} \tag{2.9}$$

where the $\frac{2K_p+K_iT}{2}$ term can be called B0 and $-\frac{2K_p-K_iT}{2}$ can be called B1. After cross-multiplication and solving for PIout[0], the equation becomes

$$PIout[0] = B0Y[0] + B1Y[1] + PIout[1] \tag{2.10}$$

where PIout[1] is the past calculated value of the PI loop in the last iteration of the PLL. The PIout term is called ylf in code and the K_p and K_i terms were calculated using the spll_coeff_compute.xlsx spreadsheet provided by the example files in the ControlSuite folder that comes with the TI Code Composer Studio compiler for C coding of the microcontroller [16]. The first three steps in C code is shown below with the transfer function equation solved for the output of each stage [16].

Listing 2.2: First three steps of the PLL

```
//***************************************************
//Phase Detect
//***************************************************
spll_obj->Upd[0]=spll_obj->AC_input*spll_obj->cos[1];

//***************************************************
//Band-pass Filter
//Gets rid of twice the frequency in the voltage
//***************************************************
spll_obj->ynotch[0]=
(spll_obj->notch_coeff.B0_notch*
   (spll_obj->Upd[0]+spll_obj->Upd[2]))+
   (spll_obj->notch_coeff.B1_notch*spll_obj->Upd[1])-
  (spll_obj->notch_coeff.A1_notch*spll_obj->ynotch[1])-
     (spll_obj->notch_coeff.A2_notch*spll_obj->ynotch[2]);
//***************************************************
//Low Pass Filter
//PI Controller that filters out harmonics of Vpcc
//***************************************************
spll_obj->ylf[0]=spll_obj->ylf[1]+
  (spll_obj->lpf_coeff.B0_lf*spll_obj->ynotch[0])+
  (spll_obj->lpf_coeff.B1_lf*spll_obj->ynotch[1]);
  spll_obj->wo=spll_obj->wn+spll_obj->ylf[0]; //update the output
     frequency in w (Q2PI*f)
```

The last stage is the VCO where the sinusoidal instantaneous output of the inverter is calculated. The filtered phase shift, from the previous stages, is first added to the angular frequency and

then converted to degrees. The algorithm then checks whether the calculated theta is greater than or equal to 2π and wraps the theta value around to 0 if true. The phase shift from the inverter LPF is also taken care of in the PLL routine. The new theta (θ) is then used as an index to lookup the corresponding cos(θ) and $sin(\theta)$. In order to save memory, the stored lookup table only contains the first quadrant of $sin(\theta)$ values ($sin(0)$ to $sin(\frac{\pi}{2})$). To calculate the other quadrant and cosine values, simple trigonometric identities were used as shown in the code below. The PLL code calculations are kept in floating point because the incoming ADC samples are normalized from -1 to 1 and simpler conversion between the theta value to array index in the lookup table.

Listing 2.3: Sine wave generation portion of PLL

```
//---------------------------------//
// VCO
// Internal voltage oscillator that changes based on the error from
   PLL
//---------------------------------//
degreeDesired=13.5 + phaseshift;

float newTimeShift=(degreeDesired*PI)/180; //convert the desired
   phase shift to radians


//compute theta value
spll_obj->theta[0]=spll_obj->theta[1]+
(spll_obj->wo*spll_obj->delta_t);
if(spll_obj->theta[0]>(float)(6.2832)){ //greater than 2 pi
  spll_obj->theta[0]=(float)(0.0);
}
int8_t sineSign=1;
 if(spll_obj->theta[0] > (QPI2) && spll_obj->theta[0]< (PI)){ //Q2
    sinTheta=PI-spll_obj->theta[0];
  }

 if(spll_obj->theta[0]> PI && spll_obj->theta[0]< (Q3PI2)){ //Q3
     sinTheta=spll_obj->theta[0]-PI;
```

```
   sineSign=-1;
}
if(spll_obj->theta[0]> (Q3PI2) && spll_obj->theta[0] < (Q2PI)){ //Q4
    sinTheta=(Q2PI)-spll_obj->theta[0];
    sineSign=-1;
}

sinTheta=sinTheta*(2/PI); // normalizing the theta
int sinIndex=(int)ceil(sinTheta*(1023)); //translating the theta
    into an index value
  spll_obj->sin[0]= sineSign*sineTable[sinIndex]; //calculating
     sin(theta)*1 or sin(theta)*-1 dependent on the quadrant sign

  sineSign=1;
float cosIndex=spll_obj->theta[0]+(QPI2); //changing the theta for
    cos and repeating the process
```

Instead of outputting the internal sinusoidal wave to the inverter SPWM routine, the output of the PLL is the internal sinusoidal wave with an added phase shift. The additional phase shift is necessary to set the real power output of the inverter. Currently, the additional phase shift in this work is set to 0. At the end of the PLL subroutine, the old values of the filter, sine/cosine wave, and theta are all updated to the new calculated values.

Chapter 3

Control Algorithm

3.1 Overall Control Algorithm

The controller's overall algorithm in Figure 3.1 ties all the software modules together in a sequence of control actions the inverter takes in response to the change in the voltage at the point of common coupling (PCC). The initialization of the interrupts is completed before the program begins the main loop. At the beginning, the ADC interrupt is sampling V_{PCC} and V_{dc}. After the ADC is done with its sampling, it calls the PLL. The PLL proceeds to use the most recent sample to synchronize the internal sine wave to the exterior V_{PCC}. The SPWM ISR is also enabled and running, but m_a and phase angle of the inverter sinusoidal output is dependent on the present control mode. In real-time, the PLL takes a max of 5 seconds before synchronization is complete. Thus, a soft-start time period of 200 counts is inserted before any control action is taken.

Once the synchronization is complete, the microcontroller continuously calculates the rms values for V_{inv}, I_{inv}, V_{bat}, P_{grid}, and V_{PCC} in the main program. If V_{PCC} is greater than V_{bat}, the PWM signals are turned off and the inverter acts as a passive ac-dc rectifier. If V_{PCC} is outside V_{ref} deadband and less than V_{bat}, the voltage-reactive power mode is initiated instead. Throughout the whole program, the interrupts for the ADC, PLL, and SPWM are ongoing.

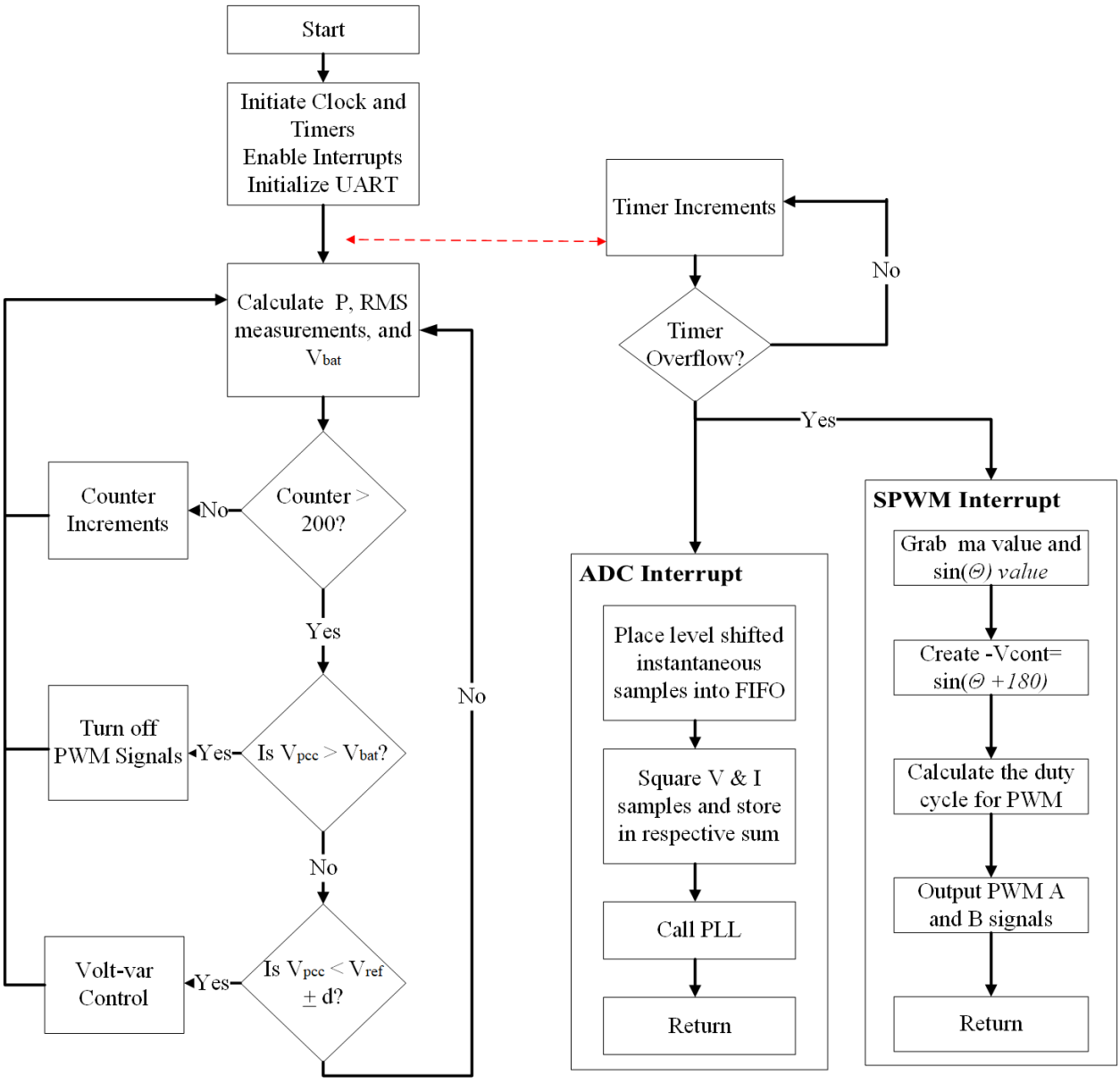


Figure 3.1: Inverter Controller Flow Diagram

The timing diagram for the interrupts is shown in Figure 3.2 where the red signal is the ADC, blue is the PLL, green is the SPWM, and the blank intervals in-between is when the main program is running after the interrupts have all returned. The timing diagram also shows the order of priority each of the ISRs have in the program. The highest priority interrupt is the ADC because the rms calculations and PLL methods depend on the ADC samples. The PLL is not an interrupt, but is called directly from the ADC ISR in order to avoid delays between the ADC sample and PLL processing. The SPWM has the lowest priority because the PWM routine will continuously output a square wave based on the previous duty cycle inputted into the PWM. Thus, the PWM gating signals do not need to be updated right away as opposed to the other two routines. All of the ISRs interrupt at 6 kHz, but the priority determines which interrupt goes first.

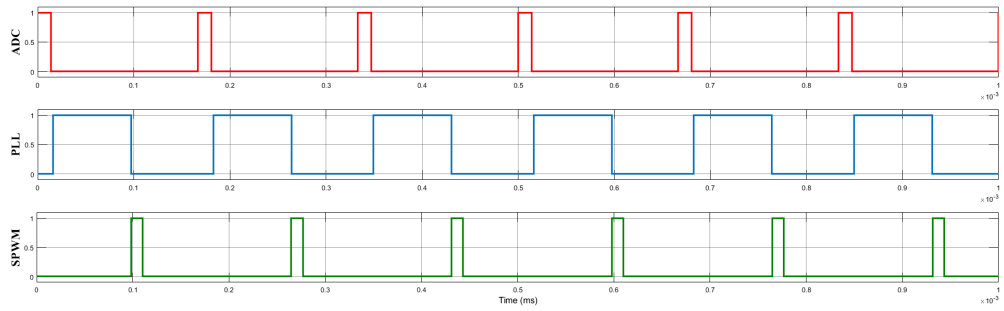


Figure 3.2: Interrupt Timing Diagram

3.2 Charging Energy Storage

As the inverter controller continuously monitors V_{PCC} , it calculates whether $V_{PCC_{peak}}$ is above V_{bat}. If the condition is true, the inverter controller shuts off the inverter PWM signals by setting m_a to zero. The SPWM interrupt is not disabled because if it is turned off, the SPWM routine would keep the old m_a value and continue to output an arbitrary PWM gating signal to the

corresponding PWM pin. Also, by not disabling the SPWM interrupt, it saves the controller from an extra step of enabling an interrupt during the transition between the charging and discharging modes. Once m_a is set to zero and $V_{PCC_{peak}}$ is higher than V_{bat}, the battery begins to passively trickle-charge. When $V_{PCC_{peak}}$ drops below V_{bat}, the controller re-enables the PWM signals and performs volt-var control when necessary. The experimental grid voltage is kept within the battery's charging limits, so battery protection was not considered in this work.

3.3 Voltage and Active Power Regulation

The equation set 3.1 contains the power flow equations for when an inverter, load, and grid are connected in parallel to the same PCC node as shown in Figure 3.3. V_{PCC} and X_s (the coupling inductance on the inverter) are out of control of the inverter, so the directly controllable parameters are V_{inv} and δ. As long as V_{inv} has a V_{dc} higher than $V_{PCC_{peak}}$, then the inverter can control the real power sent to the grid. Although both equations include $|V_{inv}|$ and δ, Q is dependent on the difference in voltage magnitude between V_{PCC} and V_{inv} while P is dependent on δ, the angle between the two voltage sources. Thus, the injected active power, P, can be controlled by δ while the injected reactive power, Q, can be controlled by $|V_{inv}|$ with the assumption that the changes in $|V_{inv}|$ and δ are small and the system is mostly inductive [17].

$$\begin{aligned} Q &= \frac{V_{PCC}(V_{inv} - V_{PCC})}{X_s} cos(\delta) \\ P &= \frac{V_{PCC}V_{inv}}{X_s} sin(\delta) \end{aligned} \tag{3.1}$$

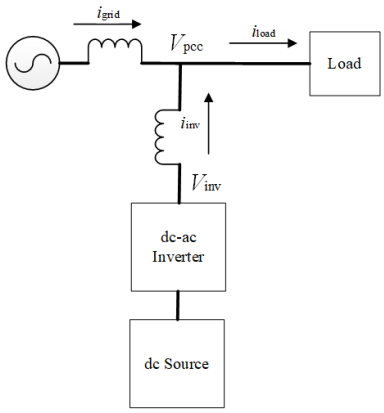


Figure 3.3: Inverter to Grid to Load Connection

3.4 Volt-Var Control Scheme

The volt-var curve presented in Figure 3.4 demonstrates which action to take in regards to reactive power [18]. When V_{PCC} is within the deadband (d) of V_{ref}, no var action will be taken. If the voltage is greater than $1.05V_{ref}$, the max Q will be absorbed. If the voltage is less than $0.95V_{ref}$, the max Q will be provided. For the values between $1.01V_{ref}$ to $1.05V_{ref}$ and between $0.99V_{ref}$ to $0.95V_{ref}$, the necessary Q to generate is calculated through linear interpolation of the slope, (m). between those points. In this work, the deadband of V_{ref} is set to be a max of $\pm 2\%$ of V_{ref} which is between 9.8 V and 10.2 V for a V_{ref} of 10 V.

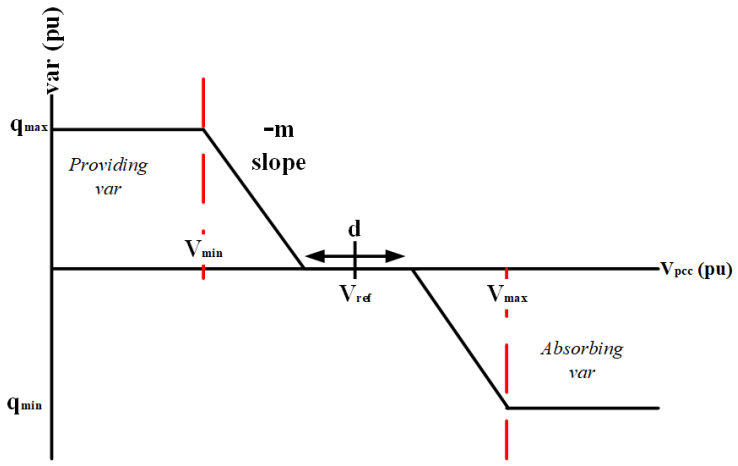


Figure 3.4: Volt-Var Curve

A different perspective of the volt-var curve is shown in Equation 3.2 where the difference in Q is a product of the slope and the voltage difference or, in other words, a proportional gain of the voltage difference. Instead of hard-coding different volt-var curves in the inverter controller, a PI control loop can be used to reduce ΔQ to zero as shown in Equation 3.3.

$$\Delta Q = m(V_{ref} - V_{PCC}) = (Q_{ref} - Q_{PCC}) \tag{3.2}$$

$$|V_{inv}| = V_{PCC} + K_p[Q_{ref} - Q_{meas}] + K_i \int_0^t [Q_{ref} - Q_{meas}]dt \tag{3.3}$$

Thus, the control algorithm does not need to explicitly calculate the required reactive power to supply. Per Equations 3.2 and 3.3, the new equation can be written as

$$|V_{inv}| = V_{PCC} + K'_p[V_{ref} - V_{PCC}] + K_i \int_0^t [V_{ref} - V_{PCC}]dt \tag{3.4}$$

where K'_p incorporates the volt-var slope, m, and the conversion factor between ΔQ and the inverter voltage magnitude $|V_{inv}|$. K_i also reduces any steady state errors by keeping a running history of the past PI control actions. Any inverter losses are also taken care by the PI controller

since circuit losses will cause an error between V_{ref} and V_{PCC}. One important note for the integral error is that the most recent error should be scaled by K_i and then integrated to avoid a spike added to the integral error [19]. The result of the PI is then added to the current V_{PCC} measurement to calculate the necessary $|V_{inv}|$ to achieve the target V_{PCC}. Because $|V_{inv}|$ is in rms, the magnitude needs to be multiplied by $\sqrt{2}$ to convert it into peak voltage for direct comparison with V_{bat}. $|V_{inv_{peak}}|$ is divided by V_{bat} to calculate the amplitude ratio, m_a. As discussed earlier, the calculated ratio is sent to the SPWM control signals which allows the inverter to output an ac voltage at a fraction of the full output. To avoid fluctuations in m_a, a moving average filter is applied to the past 10 m_a values. Figure 3.5 summarizes the volt-var control algorithm in a block diagram.

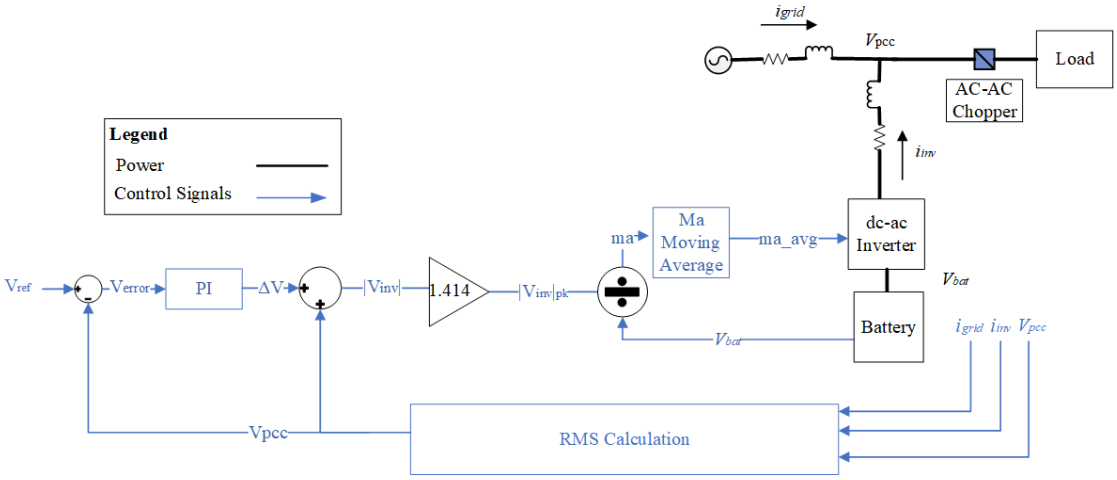


Figure 3.5: Volt-Var Block Diagram

In order to tune the PI control parameters, the Thevenin equivalent of the grid and the inverter was simulated in Matlab Simulink with a variable resistor as the varying load. Without knowing the system's transfer function, the simulation of the experiment can be used to tune the PI controller within the volt-var control loop. Initially in the tuning process, K_i is set to 0 and

K_p is doubled until the volt-var method gradually settles slightly below the setpoint [20]. After several trial and error runs, the best K_p was at 4. Then, the integral term was incremented until the steady-state error reached 0 at K_i=1.5.

Chapter 4

Experimental Results

In this chapter, the verification of each inverter module (SPWM, rms calculations, and PLL) is discussed. Before any overall control action can be implemented, the inverter controller needs to have its modules working properly. After verifying the modules, the tabletop distribution grid model configuration and the experimental parameters are described. Lastly, the results of the volt-var and battery charging tests are presented and analyzed.

4.1 Sinusoidal Pulse-width Modulation Verification

The first step in constructing the controllable inverter is verifying that the inverter has the ability to output a voltage sinusoidal wave. This test will verify whether the inverter can sample an incoming sinusoidal signal, create its inverse, and perform unipolar SPWM as described in Section 2.3. For the experiment, the microcontroller directly samples the incoming V_{PCC} voltage from the SCB output, generates -V_{PCC} sample by sample, and calculates the duty cycle depending on the current comparison between V_{PCC} and the timer counter. This duty cycle is then fed into the PWM ISR to output A and B gating signals that can replicate the incoming sine wave. Figure 4.1 shows the results of the SPWM code where the orange signal is V_{ab} control signal and the purple signal is the original V_{in} signal. The test was successful because V_{ab} is in phase with the incoming sine wave. The MOSFETs in the H bridge replicate the switching pattern of the orange square wave

and the output LPF filters out the harmonics to output a sinusoidal signal duplicate to the original.

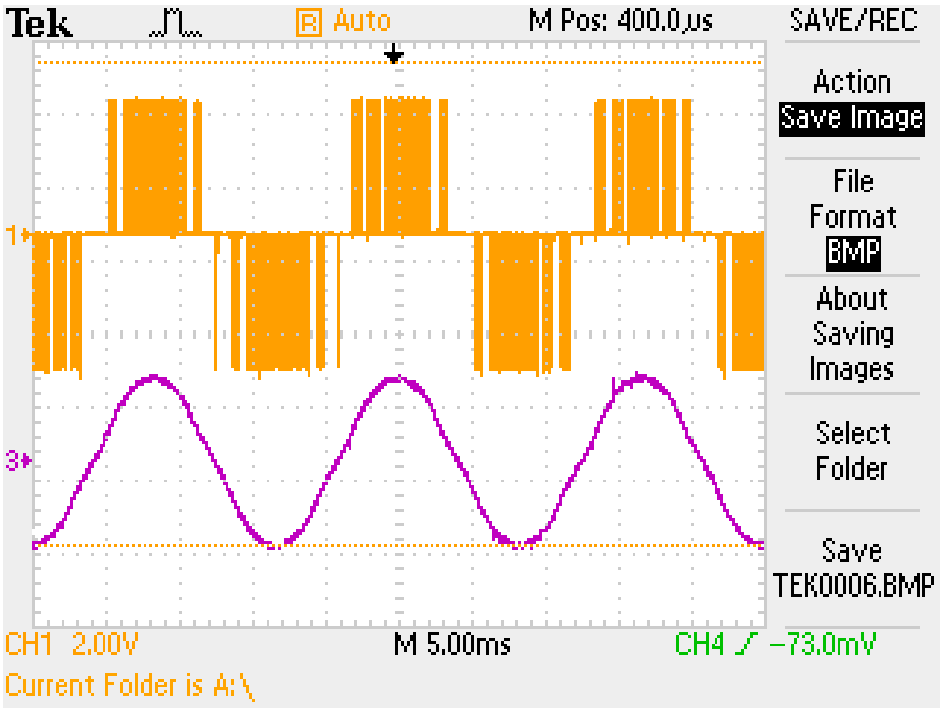
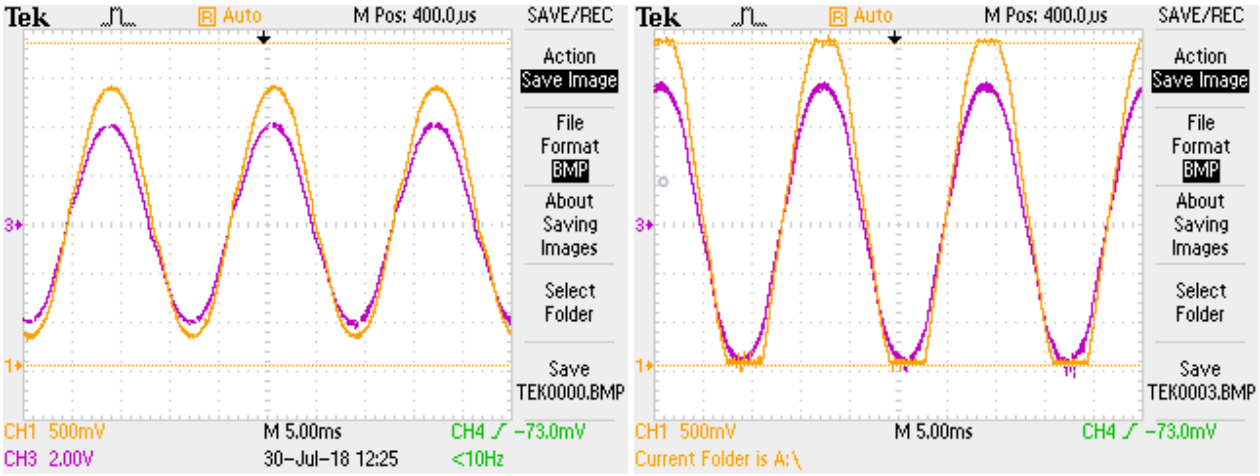


Figure 4.1: SPWM Signals and the Input Sine Wave

4.2 SCB Verification and RMS Calculation

This section describes the testing of the SCB design and the accuracy of the rms calculations in code. Before connecting the SCB output to the microcontroller, the input voltage samples must be conditioned to prevent overvoltage to the ADC ports. The SCB needs to be verified whether it can replicate V_{in} phase information, provide a proportionally lower magnitude of V_{in}, and maintain the signal voltage level to be within 0 to 3.3 V. In this test, V_{PCC} is measured and calculated. In Figures 4.2a and 4.2b, the orange signal is the SCB output while the purple signal is the original V_{in}. Figure 4.2a verifies that the board successfully level shifts and shrinks the in-



(a) (b)

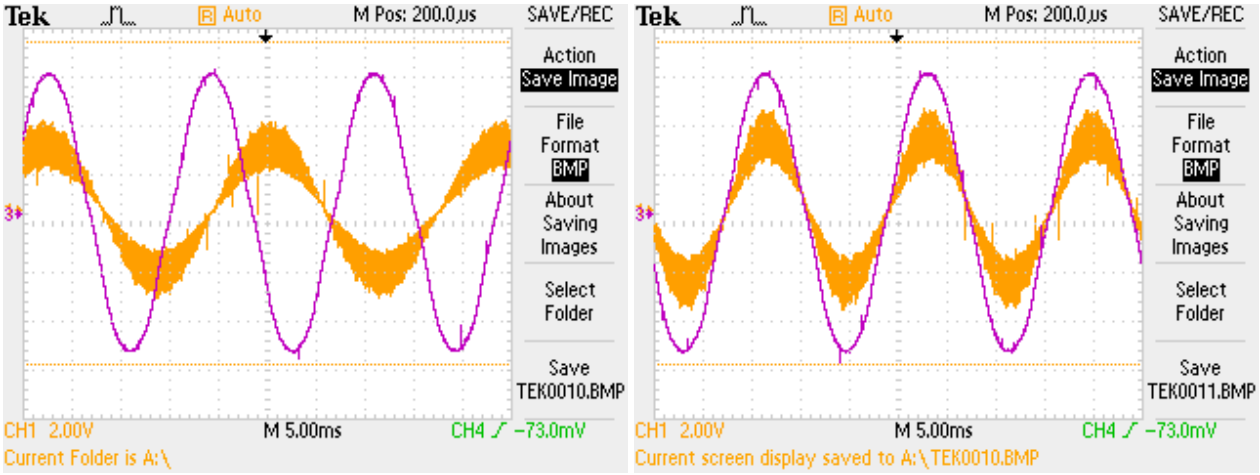
Figure 4.2: Signal Conditioning Board Results

coming sinusoidal to be within 0 and 3.3 V. The orange horizontal lines indicate the 0 V and 3.3 V boundaries. If the transformed waveform exceeds this voltage limit, the waveform is saturated at the voltage boundaries as shown in Figure 4.2b. Thus, the voltage divider ratio plays a critical role in ensuring the incoming ac signal stays within 1.65 to -1.65 V and not saturate later. If the SCB output is saturated, there will be errors in the rms calculation. This test also proves the microcontroller will not be destroyed by an overvoltage regardless of voltage spikes or the input ac signal magnitude.

After the SCB output is verified, the inverter controller proceeds to verify the $V_{PCC_{rms}}$ calculation from the sampled waveform with the physical multimeter measurement. Inaccurate measurements and calculations can lead to incorrect control actions, so it is important that the rms calculations match the physical measurement. The $V_{PCC_{rms}}$ calculations were within $\pm 1\%$ error in comparison with the multimeter measurements.

4.3 Phase Locked Loop Verification

In this test, the PLL is verified to determine whether the microcontroller can synchronize the inverter output voltage with the grid voltage before grid tie. Figures 4.3a and 4.3b compares the inverter output voltage in orange and V_{PCC} in purple. Without the PLL, the inverter voltage is out of phase with respect to V_{PCC} as shown in Figure 4.3a. Figure 4.3b showcases the PLL successfully synchronizing both voltage waveforms together. In the SPWM test, the inverter had no control over $|V_{inv}|$ or δ between the two voltage waveforms. With a successful PLL, the inverter voltage magnitude and phase difference of the inverter can be altered elsewhere in the algorithm like during volt-var control.



(a) (b)

Figure 4.3: Before and After PLL Synchronization

4.4 Distribution Grid Setup

The distribution grid setup for the overall inverter testing is shown in Figure 4.4. The incoming grid voltage is connected to a 240/24 V auto-transformer and in parallel with both the inverter and varying ac load. Both voltage sources feed into the PCC which is connected to an ac-ac chopper that controls the voltage into the load. The ac/ac chopper is a converter that decreases the input ac voltage to a specified ac voltage dependent on the duty cycle given to the load controller [21]. By providing an array of duty cycles, the chopper can be used to simulate various load profiles. On the inverter side, two 12 V, 12 Ah batteries connected in series are used for V_{dc}. A grid-tied switch is inserted between the inverter and grid, so the inverter could synchronize with the grid voltage before the switch is closed. Due to the availability of existing parts in the lab, the inverter was built with reused FDA59N25 MOSFETs and 10 A fuses from past experiments. The main voltage limitation on the inverter is the rated dc 250 V MOSFETs and the ac current limitation is from the fuses rated at 10 A [14]. The rated apparent power (S_{rated}) for the inverter is 1.768 kVA while P_{max} is 169.8 W due to the 24 V input dc voltage source and the 10 A fuses. However, it is best to operate the inverter at 1.25 kVA where the max input dc voltage is 125 V. This is to avoid exceeding the MOSFET voltage rating during transient voltage spikes that can happen when the MOSFET is switching due to the inductive filter trying to resist the change in current. At the current dc voltage selected for this work, the inverter has enough capacity for Q control and any future P control as well since 1.25 kVA is larger than 169.8 W.

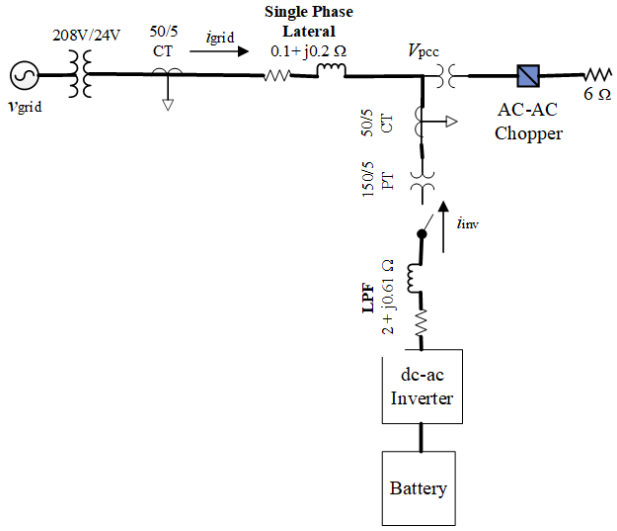
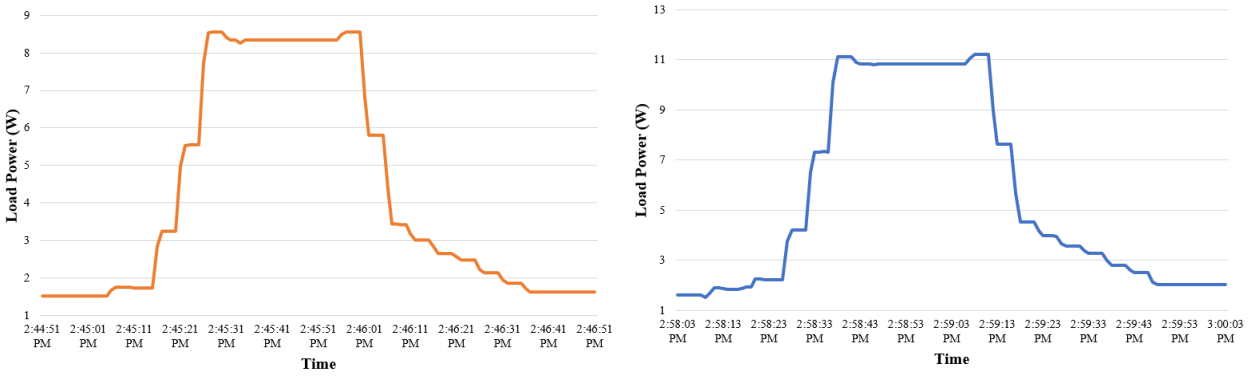


Figure 4.4: Distribution Grid One-line

4.5 Volt-Var Test

In this experiment, the power inverter is regulating the point of common coupling voltage to a target deadband of $\pm 2\% V_{ref}$. After further PI parameter tuning on the physical system, K_i was found to differ slightly from the simulation results. When running the physical experiment, K_i of 1.5 increased the initial oscillation and rise time in the PI control response. K_i was then gradually decreased until K_i was at 1. With the same V_{ref} at 10 V, two load profiles were run to showcase volt-var control.

The first load profile is shown in Figure 4.5a where the load demand varies from 1.5 W to 8.5 W. The second load profile is shown in Figure 4.5b where the load demand varies from 1.62 W to 11.2 W. The waveshape of both load profiles are the same because they both use the same array of duty cycles in the ac-ac chopper, but the power demand is different.



(a) (b)

Figure 4.5: Undervoltage and Overvoltage Load Profiles

The load profile in Figure 4.5a varied V_{PCC} from 8.52 V to 9.73 V for 2 minutes without regulation as shown in Figure 4.6 from 7:15:29 PM to roughly 7:19:59 PM. By running the curve under the desired V_{ref}, the experiment tests whether the controller can regulate the undervoltage case. As seen in Figure 4.6, after 14 seconds of settling time, the controller regulated the voltage roughly within the set deadband from 7:20:29 PM to 7:22:59 PM. In the second load profile, V_{PCC} varied from 10.09 V to 11.44 V from 7:23 to 7:25 PM. As shown, the controller can regulate the over voltage case during steady-state as well.

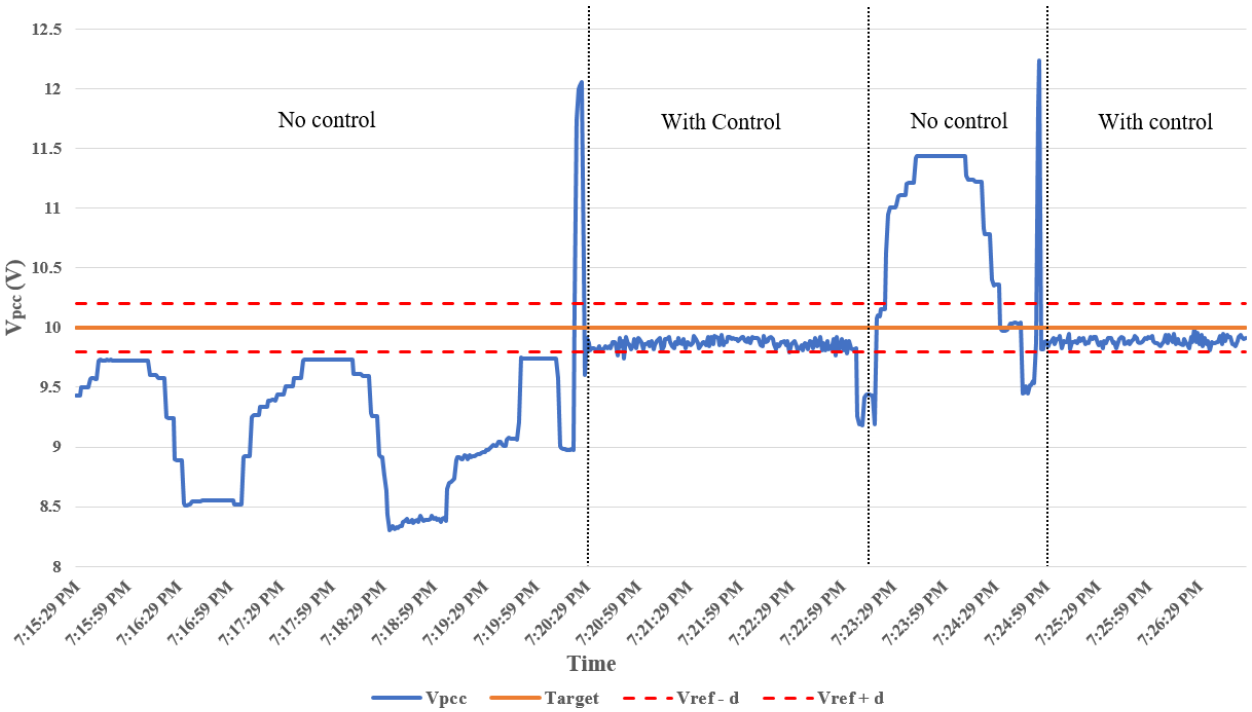


Figure 4.6: Result of Volt-Var Test with V_{ref}=10 V

A closer look at the effects of the PI control response is shown in Figure 4.7. There are two problems evident from the graphs: the initial overshoot and the steady-state voltage oscillation. After multiple trials, the first problem happens when there is a large difference in magnitude between V_{PCC} and V_{inv} and the grid-tied switch is open. Due to the voltage difference between V_{inv} and V_{PCC}, there is an inrush current from the inverter to the load. The initial overshoot is from the inverter's LPF inductor's opposition to the change in current when the grid-tie switch is manually switched on, causing an overvoltage. If V_{PCC} and V_{inv} are at the same magnitude, the change in current is reduced along with the voltage spike. The voltage spike was avoided when the grid-tied switch was closed after V_{inv} synchronized with V_{PCC} and volt-var control had not started yet.

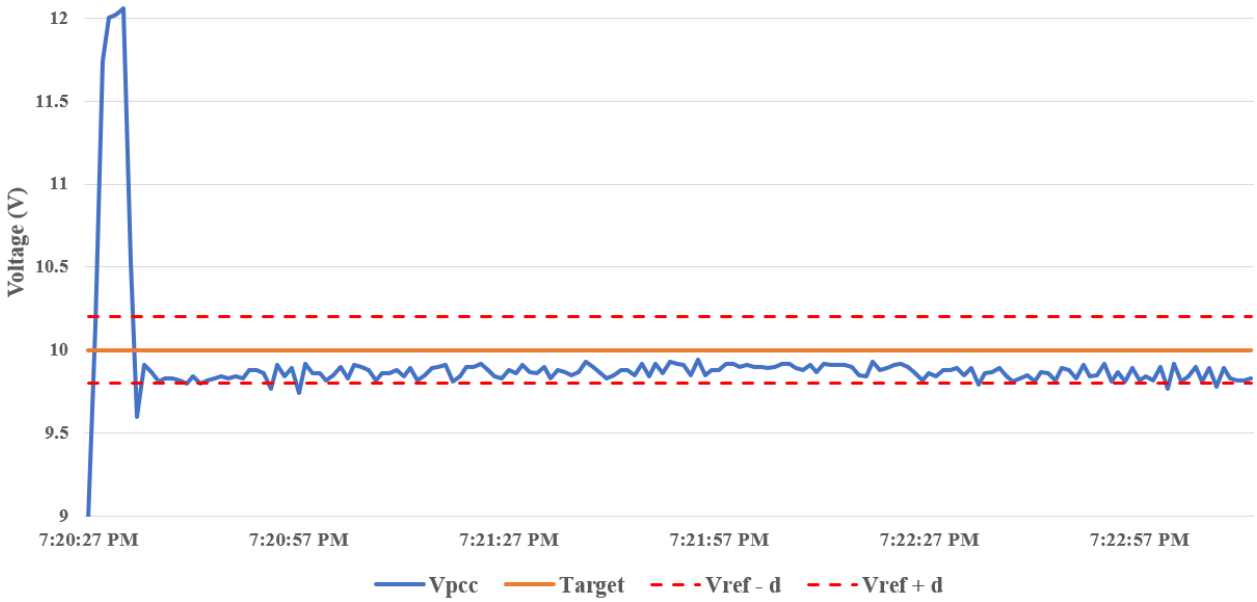


Figure 4.7: Closer Look at the Volt-Var PI Response for V_{ref}=10 V

In the future, the manual grid-tied switch can be replaced with a transistor and closed during the zero crossing of V_{PCC} to avoid this problem. The second problem of the steady-state voltage oscillation is due to the $\pm 1\%$ error in the voltage rms calculation. Since the rms voltage calculation is fluctuating, the PI controller constantly attempts to correct V_{PCC} causing a steady-state oscillation around the V_{ref} deadband. Originally, the problem was attributed to the PI controller parameters, but because the oscillation did not have a repetitive pattern, the idea was abandoned. The volt-var control results satisfy the conditions listed in Table 1.1 because the steady-state voltage stayed within $\pm 2\% V_{ref}$ and reached the target in less than 90 s. Lastly, the mode can be turned on/off which is demonstrated in the next section.

4.6 Battery Charging

In order to keep the grid voltage low and the load heat dissipation to a minimum, one 12 V battery was used to test the charging capabilities of the inverter. The inverter's LPF 2 Ω resistor was also replaced with a 10 Ω to limit the current into the battery. $V_{PCC_{rms}}$ was manually kept between 5 to 15.75 V in order to showcase the inverter performing volt-var up to 8 V and the battery starting the charging mode at 12.9 V. When the voltage is higher than $\frac{V_{dc}}{\sqrt{2}}$, the volt-var control is disengaged and the inverter gating signals are off. The inverter controller then waits for the grid voltage to rise high enough to charge the battery. Once $V_{PCC_{peak}}$ falls below V_{bat}, the PWM signals are re-engaged and the inverter performs volt-var control when V_{PCC} is outside the $\pm 2\% V_{ref}$ deadband.

Figure 4.8 shows the $V_{PCC_{rms}}$ in dark blue, V_{bat} in light blue, and $I_{invpeak}$ in orange. At the end of the graph, during the battery discharging section when volt-var control is engaged, there is a dotted dark red reference line to showcase the target V_{ref} which is set at 5 V in this test.

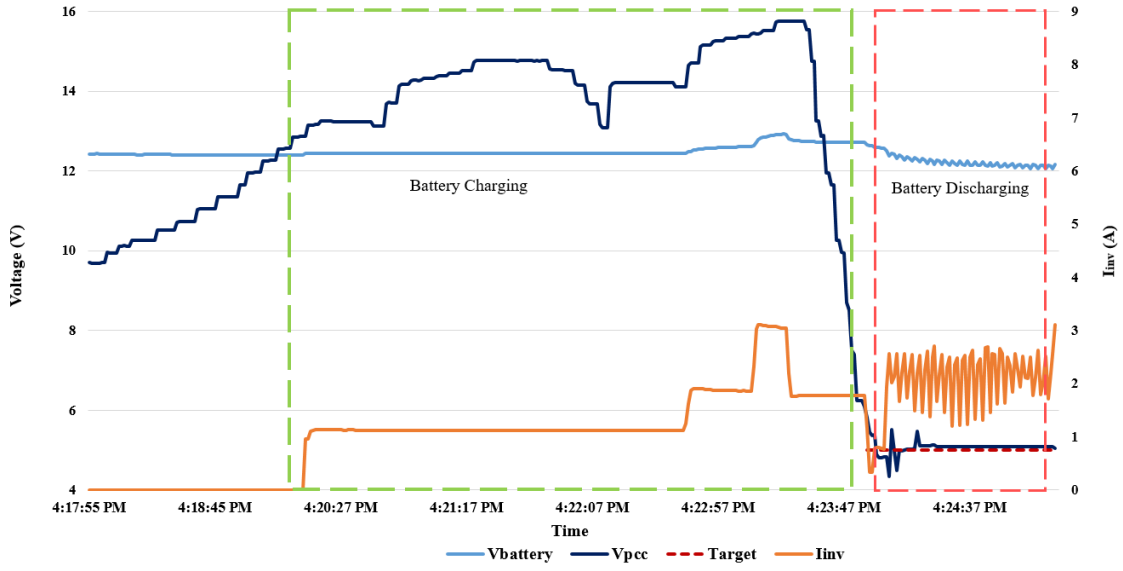


Figure 4.8: Switching Between Charging/Discharging Modes

The figure has four sections where the first section is the battery during idle mode and not performing any action to the distribution grid model. At roughly 4:20 PM, $V_{PCC_{peak}}$ is gradually increasing above V_{bat}. The battery begins to trickle-charge from the incoming I_{grid}. The varying $V_{PCC_{rms}}$ during the battery charging state is due to the ac-ac chopper repeating the load profile shown in Figure 4.9 where the load demand varies from 3.83 W to 25.1 W. As expected, the higher I_{grid} charges the battery quicker as observed from 4:22 PM to 4:23 PM in Figure 4.8. At roughly, 4:23:47 PM, $V_{PCC_{rms}}$ starts to drop back below 8.48 V. During this transition time between the battery charging and discharging, no control action is taken since the inverter only takes action when $V_{PCC_{rms}}$ is less than $\frac{V_{dc}}{\sqrt{2}}$. The battery is discharging at 4:24 PM because the oscillating current is from the inverter absorbing and supplying reactive power to regulate the voltage back to V_{ref}.

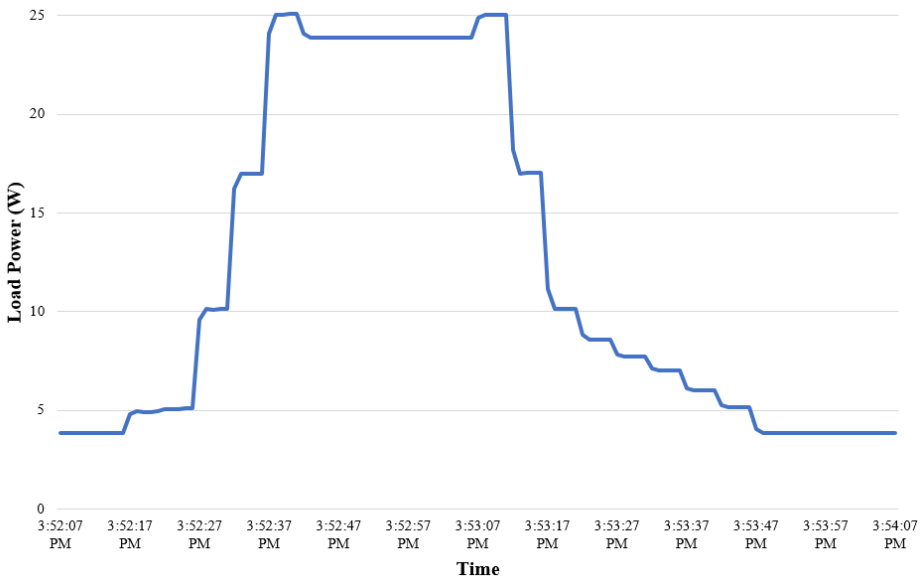


Figure 4.9: Load Profile during Battery Charging

This test illustrates two points. If the voltage rise caused by the PVs is higher than the voltage of the grid-tied batteries, a bi-directional inverter can turn off, rectify the grid voltage, and absorb the reverse power flow. However, this will require the inverter to stop inverter action which can cause problems if the dc resource cannot accept reverse power flow. The other problem is that the trickle charge method is insufficient to keep the incoming current under the battery's charging current limit. With a bi-directional dc-dc converter, the converter can be controlled to protect the battery from high grid current. It can also lower grid voltage to the appropriate charging voltage level the battery requires to avoid overvoltage.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The main goal of this thesis is to design, analyze, and construct a controllable bi-directional inverter that can connect batteries to the distribution grid and perform volt-var control. In the first stage of the project, the physical H-bridge was built and the TI TM4C launchpad was selected as the inverter controller. Then, the necessary modules of a controllable inverter were identified as the measurement circuitry, grid synchronization code, and the sinusoidal PWM gating signals. Each module was designed and validated individually. They were then combined together into an overall control algorithm that can perform volt-var regulation and toggle between the battery's charging/discharging modes.

Next, the overall bi-directional inverter volt-var control and the battery charging mode were verified. For the volt-var regulation, the steady-state voltage results stayed within 2% of V_{ref} and reached settling point within 90 s as intended in the objectives listed in Table 1.1. For the bi-directional power flow test, the battery successfully charged up to 12.9 V and discharged during volt-var regulation. However, it is apparent in the results that a current limiter between the battery and inverter is necessary to protect the battery in future experiments with higher voltage operating points.

5.2 Future Work

As mentioned above, the charging capability of the battery is limited due to the lack of a current limiter that can control the charging current of the battery. For future work, a bi-directional dc-dc converter can be inserted between the battery and the inverter to allow for better control over the battery's current as well as the charging voltage level. Also, the inverter's voltage output is limited by the max capacity of the battery, but the additional bi-directional dc/dc converter can increase the V_{dc} without have to upgrade the dc resources. Future researchers can also use the inverter and controller as a starting platform to test additional smart inverter features like voltage-real power support and balancing the resources between real power and reactive power support. One of steps that consumed the most time in this work was manually tuning the PI controller, so another improvement to the project can to incorporate adaptive PI gains into the control algorithm to avoid additional tuning whenever the system changes. Lastly, the inverter can be deployed in a microgrid model to test intentional islanding scenarios or as an inverter between the distributed energy resources (DER) in a DER management system.

Bibliography

[1] NCSL, "State renewable portfolio standards and goals," 2018, http://www.ncsl.org/research/energy/renewable-portfolio-standards.aspx, Last accessed on 2018-07-30.

[2] Solar Energy Industries Association, "US solar market adds 2.5 GW of PV in Q1 2018, growing 13% year-over-year," June 2018, https://www.seia.org/news/us-solar-market-adds-25-gw-pv-q1-2018-growing-13-year-over-year, Last accessed on 2018-07-30.

[3] J. von Appen, M. Braun, T. Stetz, K. Diwold, and D. Geibel, "Time in the sun: The challenge of high PV penetration in the German electric grid," *IEEE Power and Energy Magazine*, vol. 11, no. 2, pp. 55–64, March 2013.

[4] E. Stewart, J. MacPherson, S. Vasilic, D. Nakafuji, and T. Aukai, "Analysis of high-penetration levels of photovoltaics into the distribution grid on Oahu, Hawaii," *Contract*, vol. 303, pp. 275–317, 2013.

[5] N. Wade, P. Taylor, P. Lang, and J. Svensson, "Energy storage for power flow management and voltage control on an 11kV UK distribution network," in *CIRED 2009 - 20th International Conference and Exhibition on Electricity Distribution - Part 1*, June 2009, pp. 1–4.

[6] "Inverter control," Toshiba, 2018, https://toshiba.semicon-storage.com/ap-en/design-support/e-learning/mcupark/inverter-control-1.html, Last accessed on 2018-07-30.

[7] A. Namboodiri and H. S. Wani, "Unipolar and bipolar pwm inverter," *IJIRST –International Journal for Innovative Research in Science Technology*, vol. 1, no. 7, pp. 237–243, December 2014.

[8] *ARM cortex-M4F based MCU TM4C123G launchpad*, Texas Instruments, 06 2014.

[9] O. Wong, K. Chandra, V. Joshi, J. Heider, S. Maity, G. Maples, S. Jothibasu, and S. Santoso, "Simplified benchtop model of a distributed energy resource management system," in *2018 IEEE Power and Energy Society General Meeting (PESGM)*, 2018.

[10] *NMH1212SC Isolated 2W Dual Output DC/DC Converters*, Murata, 2017.

[11] *TL08xx JFET-Input Operational Amplifiers*, Texas Instruments, 02 1977.

[12] P. FUCHS, "Digital power and energy meter with DSP TMS320C6711," University Lecture, 2006, http://www.ti.com/lit/ml/sprp504/sprp504.pdf, Last accessed on 2018-07-30.

[13] M. Flynn, "Lab week 9 EE462L PWM inverter control circuit.pdf," 2016. [Online]. Available: https://utexas.instructure.com/courses/1158838

[14] *FDA59N25 N-Channel UniFET MOSFET*, Fairchild, 04 2014.

[15] *IRS21844 Half Bridge Driver*, Infineon Technologies AG, 12 2006.

[16] M. Bhardwaj, "SPRABT3A Software Phase Locked Loop Design Using C2000™ Microcontrollers for Single Phase Grid Connected Inverter," Texas Instruments, Tech. Rep., 07 2013.

[17] Binh, Tran Cong and Dat, Mai Tuan and Dung, Ngo Manh and An, Phan Quang and Truc, Pham Dinh and Phuc, Nguyen Huu, "Active and reactive power controller for single-

phase grid-connected photovoltaic systems," 2009, https://pdfs.semanticscholar.org/4265/28c4cabfee9322583f0c682d5d0ccf4b44ef.pdf, Last accessed on 2018-07-30.

[18] A. Singhal, V. Ajjarapu, J. C. Fuller, and J. Hansen, "Real-time local volt/var control under external disturbances with high PV penetration," *IEEE Transactions on Smart Grid*, pp. 1–1, 2018.

[19] Jason Sachs, "How to build a fixed-point pi controller that just works: Part I," 2012, https://www.embeddedrelated.com/showarticle/121.php, Last accessed on 2018-07-30.

[20] F. Haugen, "The good gain method for PI (D) controller tuning," *Tech Teach*, pp. 1–7, 2010.

[21] S. Jothibasu and M. K. Mishra, "A ac-ac converter based topology for mitigation of voltage sag with phase jump," in *2013 IEEE 8th International Conference on Industrial and Information Systems*, Dec 2013, pp. 259–264.